High Accuracy and Low Regret for User-Cold-Start Using Latent Bandits

David Young, Douglas Leith

School of Computer Science and Statistics, Trinity College Dublin, Ireland

Abstract. We develop a novel latent-bandit algorithm for tackling the cold-start problem for new users joining a recommender system. This new algorithm significantly outperforms the state of the art, simultaneously achieving both higher accuracy and lower regret.

1 Introduction

In a recommender system, when a new user joins the system it initially has no knowledge of the preferences of the user and so would like to quickly learn these[1]. The recommender system therefore initially starts in an "exploration" phase where the first few items that it asks the new user to rate are chosen with the aim of discovering the user's preferences. We focus on the simplest setup where a user explicitly rates items presented to them, e.g. on a 1-5 scale or binary like/dislike feedback, and the aim of the recommender system is to predict other items that the user may like.

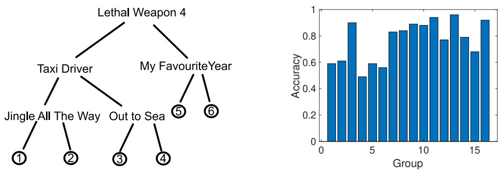
One common approach to this new user cold-start task is to take ratings already collected from a population of users, use these to cluster users into groups and then train a decision-tree to learn a mapping from item ratings to the user group, see for example Figure 1(a). When a new user joins the system this decision-tree is used to decide which items the user is initially asked to rate and in this way the group to which the user belongs is initially estimated. Once the group is estimated, the system recommends items liked by members of that group e.g. using matrix factorisation or another collaborative filtering approach.

However, typically users clustered in the same group do not give identical ratings to an item. Rather there is a spread of ratings, and this intra-cluster variability between users can be thought of as adding noise to the ratings. Decision-trees are vulnerable to such noise in the new users rating as an unusual rating for a particular group can send the tree down a wrong path it will never recover from. For example, Figure 1(b) shows the measured decision-tree accuracy for Netflix data clustered into 16 groups (see later for details). It can be seen that the accuracy is as low as 50-60% for some groups.

In this paper we improve on this behaviour by developing a new online learning algorithm that maintains and updates a probability distribution for the users group, and then selects the items that a new user is asked to rate so that this distribution converges to correct group as quickly as possible. In this way for users with a particularly large amount of noise in their ratings it will simply take the system longer to learn the correct group, as opposed to the system possibly concluding the wrong group as when using a fixed decision tree.

To develop our new online learning algorithm we view the cold-start task as a latent bandit, i.e. a multi-arm bandit where the distribution of the arms depends on the value of someone unknown latent parameter. In our setting the arms are the available items, the reward of an arm is the user rating for the corresponding item and the latent variable is the true group of the user. It is important to stress that ignoring the latent groups and applying standard bandit algorithms to this task leads to poor performance since (i) there are many arms and so learning is slow and (ii) repeated

[1] The system may have some general context regarding the user, such as the country/city they are located in, user gender and demographics *etc*, in which case learning is conditioned on this but the fundamental cold start task otherwise remains unchanged.



(a) (b)

Fig. 1: (a) Illustrating a movie recommender decision-tree (adapted from [1]), (b) Decision-tree accuracy for Netflix data (16 groups).

pulls of the same arm tend to be highly correlated[2]. In contrast, the latent group can take a small number of values, e.g. there might only be 16 or 32 groups, and for each value there is a known distribution of arm rewards. The latter means that we do not need to pull every arm to gain information about its reward and so the latent group approach allows fast learning even when the number of arms is large.

Further, in general, some arms tend to be more informative than others for learning the user group. Hence, pulling only the most informative arms allows us to quickly learn the user group. Importantly, provided we have a sufficient number of informative arms then fast learning can still be achieved even when each arm is only pulled once i.e. a new user is only asked to rate any given item once. Informative arms tend to have distinct ratings by users in different pairs of groups. We want to select the next arm to pull based on our current estimate of the probability distribution for the users group and with the aim of causing our estimated distribution to quickly concentrate on the true user group. Note that existing methods for latent bandits require repeated arm pulls and do not take full advantage of the informative arms.

2 Related Work

We follow on from the work of [2] in using cluster based bandits to tackle the cold start problem. In that paper the focus is on finding and using distinguishing items to learn, and then exploiting once a certain threshold has been passed. We approach the problem as that of a latent bandit with known reward distributions for each possible value of the latent parameter. This bears most similarity to the work of [3], which develop algorithms based on UCB and Thompson sampling for the same setup, and is the first paper to factor in model misspecification. Our algorithm however leverages the latent-bandit in a way there methods do not. In [4], they first apply a UCB style algorithm to the same problem, before relaxing assumptions and proposing algorithms for problems where the reward distributions are unknown. The information leveraging approach of our algorithm is most related to the work of [5], which proposes a heuristic for choosing between policies so as to balance information gain with regret. However, as they note in their paper, for our setup their approach results in an intractable integral that cannot easily be solved or approximated. A survey of active learning cold-start methods can be seen in [6], and both [7] and [8] use decision tree based methods in a similar cluster based setup to ours.

3 Latent Bandit Algorithm For User Cold-Start

We have a set $\mathcal{G}$ of user groups. Each user belongs to one group $g \in \mathcal{G}$. We also have a set of items V. Given a new user our task is to quickly learn which group they belong

[2]Measurement studies indicate that when people are repeatedly asked to rate the same item on a scale of 1-5 then if they rate 1 or 5 they tend to consistently stick with that rating although when they rate 3 or 4 they may change their rating back and forth between 3 and 4. That is, lumping ratings of 3-4 together in a single bucket these previous studies indicate that a user's rating of an item tends to be consistent.

to by asking the user to rate items in V, using the fact that the distribution of item ratings varies depending on the user's group.

3.1 User Ratings

We assume the rating of item v by users in group g is normally distributed with mean $\mu(g,v)$ and variance $\sigma^2(g,v)$. If a user u in group g is asked to rate the same item v multiple times the user gives the same rating i.e. their rating is one value drawn from $N(\mu(g,v),\sigma^2(g,v))$. Let random variable $R(v)$ be the rating of item v and random variable $G(u)$ be the group of the user making the rating. Then $p(R(v)=r|G(u)=g) = (1/\sqrt{2\pi}\sigma(g,v))e^{-(r-\mu(g,v))^2/2\sigma^2(g,v)}$ and for observed sequence D_n of ratings $r_1, r_2, \ldots, r_n$ for items $v_1, v_2, \ldots, v_n$ (so D_n is a sequence of pairs $(v_i, r_i), i = 1, \ldots, n$) it follows that $p(D_n|G(u)=g) = \gamma_n(g)e^{-L_n(g)}$ where $L_n(g) := \sum_{i=1}^{n}(r_i - \mu(g,v_i))^2/2\sigma^2(g,v_i)$, $\gamma_n(g) := 1/(2\pi)^{n/2} \times 1/\Pi_{i=1}^{n}\sigma(g,v_i)$. By Bayes rule, $p(G(u)=g|D_n) = \frac{p(D_n|G(u)=g)p(G(u)=g)}{p(D_n)}$ with $p(D_n) = \sum_{h\in\mathcal{G}} p(D_n|G=h)p(G(u)=h)$. Assuming uniform prior $p(G(u)=g) = 1/|\mathcal{G}|$ then the probability that a new user belongs to group g given item ratings D_n is

$$p(G(u)=g|D_n) = \frac{p(D_n|G(u)=g)}{\sum_{h\in\mathcal{G}} p(D_n|G(u)=h)} = \frac{\gamma_n(g)e^{-L_n(g)}}{\sum_{h\in\mathcal{G}} \gamma_n(h)e^{-L_n(h)}} \quad (1)$$

Note that an informative prior could be used instead of this uniform prior when, for example, contextual information such as user location, age, gender etc allows the group of a new user to be estimated in advance.

3.2 Exploration: Knowledge Gained From a New Rating

Given a new user's ratings $r_1, r_2, \ldots, r_n$ for a sequence of items $v_1, v_2, \ldots, v_n$, we need to select the next item v_{n+1} to ask the user to rate. We have a current estimate $P(G(u)=g|D_n), g \in \mathcal{G}$ of the probability distribution of the user group. Intuitively, a reasonable choice is the item that tends to cause this distribution to maximally concentrate on our current best estimate of the user group. While we might consider selecting the item v_{n+1} which causes $\mathbb{E}[P(G=g^*|D_n, v_{n+1}, R(v_{n+1}); G=g^*)$ with $g^* \in \arg\max_{g\in\mathcal{G}} P(G(u)=g|D_n)$ to increase the most. Since in reality we don't know the true group g^* we want to select the item v_{n+1} that maximises the expected value with respect to g^*, i.e.$\mathbb{E}_{g^*}[\mathbb{E}[P(G=g^*|D_n, v_n, R(v_{n+1}); G=g^*)]] = \sum_{g\in\mathcal{G}} P(G=g|D_n) \cdot \mathbb{E}[P(G=g|D_n, v_{n+1}, R(v_{n+1}); G=g)]$. This expected value cannot be found analytically, but if we take a linear approximation of $P(G=g^*|D_n, v_n, R(v_{n+1}); G=g^*)$ and take our expectation over that we obtain

$$\mathbb{E}[P(G=g^*|D_n, v_n, R(v_{n+1}); G=g^*)] \approx P(G=g^*|D_n, v_{n+1}, \mu(g^*, v_{n+1})) \quad (2)$$

Instead of the approximation (2) we could use Monte Carlo simulation to evaluate this expectation but this is considerably slower to calculate and in our tests comparing (2) with the values calculated by Monte Carlo we find that (2) has surprisingly small approximation error, and negligible effect on performance.

3.3 Exploitation: Future Reward

Selecting the next item v_{n+1} to maximise (2) prioritises learning about the group that a new user belongs to i.e. exploration. However, items which accelerate learning may attact a low user rating, and so increase regret. This is because items rated highly by members of one group tend to also be rated highly by members of at least some of the other groups, and so the user ratings for these items do not serve to strongly distinguish between groups and so allow rapid learning. We need, therefore, to balance exploration against exploitation i.e. selecting items predicted to have a high user rating.

However, when considering exploitation it is necessary to take account of the uncertainty in our current estimate of the user's group. This is because items rated highly by users in one group may not be rated highly by users of another group. Hence, if we make a mistake in our estimate of the new user's group we may end up suggesting items that attact a low rating by the user and so increase regret.

We proceed by defining the discounted future reward. For a user in group g who has already rated items $V_n = \{v : (v, \cdot) \in D_n\}$ the expected future reward is $\sum_{v \in V \setminus V_n} \mu(g, v)$, assuming they stay in the system and eventually rate all items. However, its probably more reasonable to assume there is a departure process for users, who tend to only stay in the system for some lifetime. For simplicity, we will assume the case where the departure process is modelled as an independent event after every recommendation, with constant probability β of staying in the system. We could replace this with any step dependent model for the probability of the user still being the system. We then use β to discount future rewards. Let $V^*_{n,g}$ be the sequence of items $V \setminus V_n$ sorted in decreasing order of mean rating $\mu(g, v)$. Assuming the recommender presents items to the user in this order, then the expected discounted future reward is $J_{future}(g) := \sum_{i=1}^{|V^*_{n,g}|} \beta^i \mu(g, v_{i,g})$ where $v_{i,g}$ is the i'th element in sequence $V^*_{n,g}$ and $0 \leq \beta \leq 1$ is our discount factor. And so the expected discounted future loss of acting as if the user is in group h when its actually in group g: $J_{futureloss}(g, h) := \sum_{i=1}^{|V^*_{n,g}|} \beta^i |\mu(g, v_{i,g}) - \mu(g, v_{i,h})|$ where $v_{i,h}$ is the i'th element in sequence $V^*_{n,h}$. Since we have estimates $P(G(u) = g|D_n), g \in \mathcal{G}$ of the probability distribution of the user group, we can calculate the expected discounted future regret of acting as if the user is in group h, at any step n sing: $J_{futureregret}(h) := \sum_{g \in G} P(G = g|D_n) \sum_{i=1}^{|V^*_{n,g}|} \beta^i |\mu(g, v_{i,g}) - \mu(g, v_{i,h})|$. This value is exactly the expected opportunity cost of exploiting instead of learning more information about which group the user is in.

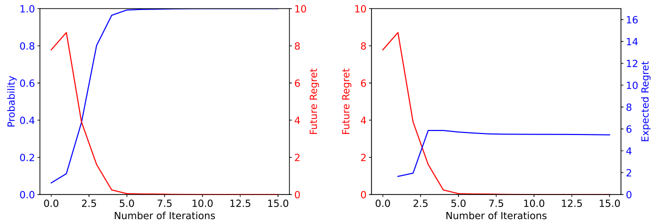
3.4 Balancing Exploration & Exploitation: New Algorithm

We balance exploration and exploitation by selecting the next item v_{n+1} that maximises

$$v_{n+1} \in \arg \max_{v \in V \setminus V_n} \sum_{g \in \mathcal{G}} P(G = g|D_n, v, \mu(g, v)) \cdot [J_{futureregret}(g) + \frac{\mu(g, v)}{J_{futureregret}(g)}] \quad (3)$$

When the opportunity cost $J_{futureregret}(g)$ of incorrectly estimating the user group is large, the short term reward $\mu(g, v)$ is effectively ignored and the next item is selected primarily to minimise $J_{futureregret}(g)$ i.e. to maximally increase the accuracy of our estimate of the user's group. Conversely, when $J_{futureregret}(g)$ is small, the next item is selected to maximise the short term reward $\mu(g, v)$ i.e. picking an item likely to be rated highly by the new user given our current best estimate of the user's group.

Figure 2(a) shows measurements illustrating the transition from exploration to exploitation for the Netflix dataset with 16 groups and a new user from group ten. It can be seen that the future regret for exploiting as if we are in group ten is initially high, as the probability for being in any particular group is low. The probability and future regret then rise together, indicating the user has given a rating that also increases the probability of being in groups for which the top rated items are very different. As the probability of being in group ten continues to rise sharply, the future regret of exploiting as if we are group ten drops sharply, as expected. Figure 2(b) shows the relation between the future regret value and the actual expected regret incurred by the system. It can be seen that the regret incurred rises steeply initially while the future regret is high, as the system focuses on learning more about the users group. Then as



(a) (b)

Fig. 2: Measurements illustrating the transition from exploration to exploitation. Netflix dataset with 16 groups and a new user from group ten. ****Add x-axes labels. DL**

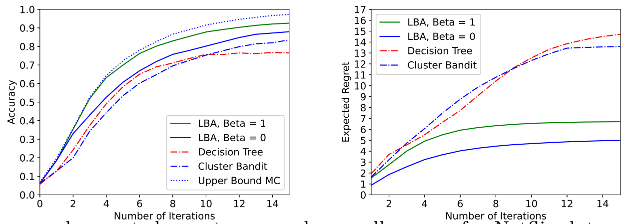


Fig. 3: Accuracy and expected regret averaged over all groups for Netflix dataset with 16 clusters. The upper bound we plot is the accuracy of our algorithm focused exclusivley on learning, with the expectation calculated using MC.

the future regret of exploiting as if we are in group ten becomes very small, the system starts to do just that, and the expected regret no longer grows.

4 Performance Evaluation

Datasets. We evaluate the performance of our latent bandit algorithm (LBA) on the standard Netflix dataset (480,189 people 17,770movies, 104M ratings from 1–5), the Jester dataset (73,421 people rating 100 jokes, 4.1M ratings from -10–10) and the Goodreads10K dataset (53,424 people rating10,000 books, 5.9M ratings from 1-5).

Clustering Users. We use training data to cluster users into groups and estimate the mean $\mu(g,v)$ and variance $\sigma(g,v)^2$ of the ratings by each group g for item v. We use the BLC matrix-factorization clustering algorithm [9] for this, although other clustering algorithms might also be used.

Baseline Algorithms. We compare the performance of the latent bandit algorithm against (i) an optimised CART decision tree and (ii) the cluster-based bandit (CBB) algorithm of [2]. These are strong baselines, with good performance for cold-start active learning. Decision-trees are often considered for use in cold-start while the recently proposed CBB-algorithm offers state of the art performance.

Modelling New Users. We generate the item ratings of a new user from group g by making a single draw from the multivariate Gaussian distribution with mean $\mu(g,v)$ and variance $\sigma(g,v)^2$ for each item equal to that estimated from the training data. This has the advantage that we can easily generate large numbers of new users in a clean, reproducible manner. In addition, we also evaluated performance when drawing ratings by splitting the data into training and test data, picking a user from the test data and using their ratings. We found the performance of these setups to be very similar to simply drawing a new user from a Gaussian distribution.

Performance Metrics. We report the accuracy with which the group of a new user is estimated, i.e. the fraction of times the correct group is estimated, and the expected regret. Statistics are calculated over 1000 new users per group.

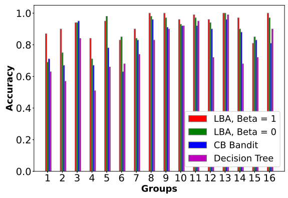


Fig. 4: Accuracy after 15 iterations vs user group. Netflix dataset with 16 clusters.

	Accuracy				Regret			
	No. of Groups				No. of Groups			
Dataset/Algo	4	8	16	32	4	8	16	32
Netflix/DT	0.92	0.9	0.77	0.55	9.6	12.1	14.8	20.2
Netflix/CB	0.96	0.93	0.89	0.74	5.1	9.8	13.9	16.6
Netflix/LBA,0	1	0.98	0.94	0.83	**2.1**	**4.3**	**5.1**	**5.5**
Netflix/LBA,1	**1**	**1**	**0.98**	**0.91**	4.1	7.2	9.7	12.4
Books/DT	0.92	0.7	0.61	0.4	3	9.5	8.4	8.4
Books/CB	0.97	0.88	0.84	0.66	5.7	8.4	9	6.1
Books/LBA,0	0.99	0.9	0.9	0.77	**1.1**	**3**	**3.6**	**3**
Books/LBA,1	**1**	**0.9**	**0.92**	**0.8**	1.1	3.5	4.7	5.2
Jester/DT	0.7	0.54	0.41	0.34	21.3	27.9	35.4	43.3
Jester/CB	0.91	0.82	0.76	0.68	15.9	22.1	33	53.8
Jester/LBA,0	0.93	0.85	0.78	0.69	**1.7**	**6.1**	**9.7**	**13.8**
Jester/LBA,1	**0.94**	**0.86**	**0.85**	**0.85**	1.8	8.9	23.4	40

Fig. 5: Performance after 25 iterations, averaged over all groups.

Figure 3 shows typical measurements of the evolution of accuracy vs #items rated by a new user. It can be seen that the accuracy grows over time and that the performance of the new latent-bandit algorithm dominates that of the decision-tree and cluster-based bandit. Data is show for future regret discount factor β of 0 and 1, when $\beta = 1$ the latent-bandit algorithm carries out purely exploration and it can be seen that, as expected, the learning rate is somewhat faster than when β is smaller, but this is balanced against increased regret (which nevertheless remains uniformly lower than that of the decision-tree and cluster-based bandit). The dotted line in left-hand plot shows the performance of the latent-bandit when g^* fixed at the correct user group. This is an upper bound on the maximum learning rate achievable by any algorithm. It can be seen that with $\beta = 1$ the latent-bandit performance is close to this upper bound, indicating little scope for further improvement. Figure 4 breaks this accuracy data down by group, observe that the variability in accuracy amongst the groups is significantly reduced by the latent bandit algorithm. Figure 5 shows summary data for the Netflix, Jester and Goodreads datasets and for 4 to 32 user groups. It can be seen that the uniformly outperforms the state of the art, simultaneously achieving both higher accuracy and lower regret.

5 Conclusion

We present a novel latent-bandit algorithm for tackling the cold-start problem for new users joining a recommender system. This new algorithm uniformly outperforms the state of the art, simultaneously achieving both higher accuracy and lower regret.

References

[1] Ke Zhou, Shuang-Hong Yang, and Hongyuan Zha. Functional matrix factorizations for cold-start recommendation. In *Proc SIGIR*, pages 315–324, 2011.

[2] Sulthana Shams, Daron Anderson, and Douglas Leith. Cluster-based bandits: Fast cold-start for recommender system new users. In *Proc SIGIR*, 2021.

[3] Joey Hong, Branislav Kveton, Manzil Zaheer, Yinlam Chow, Amr Ahmed, and Craig Boutilier. Latent bandits revisited. In *Proc NIPS '20*, 2020.

[4] Odalric-Ambrym Maillard and Shie Mannor. Latent bandits. In Eric P. Xing and Tony Jebara, editors, *Proc Int'l Conf Machine Learning*, volume 32, pages 136–144. PMLR, 2014.

[5] Daniel Russo and Benjamin Van Roy. Learning to optimize via information-directed sampling. *Operations Research*, 66:230–252, 2017.

[6] Mehdi Elahi, Francesco Ricci, and Neil Rubens. A survey of active learning in collaborative filtering recommender systems. *Computer Science Review*, 20, 06 2016.

[7] Al Mamunur Rashid, George Karypis, and John Riedl. Learning preferences of new users in recommender systems: An information theoretic approach. *SIGKDD Explor. Newsl.*, 10(2):90–100, dec 2008.

[8] Nadav Golbandi, Yehuda Koren, and Ronny Lempel. Adaptive bootstrapping of recommender systems using decision trees. In *Proc WSDM '11*, pages 594–604, 2011.

[9] Alessandro Checco, Giuseppe Bianchi, and Douglas J. Leith. Blc: Private matrix factorization recommenders via automatic group learning. *ACM Trans. Priv. Secur.*, 20(2), May 2017.